IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Fine-Tuning Of Distilbert For Gujarati-English Code-Mixed Language Identification In Resource Constrained Environment

[1]Chirag D. Shah, [2]Dr. Shailesh A. Chaudhari

[1]Assistant Professor, [2]Assistant Professor

[1]Computer Science,

[1]Smt. R. P. Chauhan Arts and Smt. J. K. Shah and Shree K. D. Shah Commerce College, KRP Darjee Indo-American Institute Of Computer Science, Vyara, India

[2]Department of IT and ICT, JP Dawer Institute of Information Science and Technology, VNSGU, Surat

Abstract: The ever-expanding use of code-mixed language, mainly on social media platforms, has resulted in challenges for natural language processing (NLP) tasks, due to its irregular writing pattern and the application of multiple languages within a sentence or phrase. Code-mixing of Gujarati-English is increasingly common in multilingual communities, as Gujarati diaspora around the world switch between their mother tongue and English while commenting/twitting/posting their views.

In this paper, we present an efficient solution for word level language identification for low resource scenarios by application of fine-tuned version of DistilBERT—a lightweight transformer-based model. Our dataset consists of code-mixed social media comments from YouTube with each word annotated as one of three language tags: Gujarati, English, or Other. It comprises of 77,761 annotated sentences containing 732,917 words, with language labels distributed as 56.06% Gujarati (GJ), 36.77% English (EN), and 7.10% Other (OT). The distinctive part of this work is its fine-tuning process which is entirely conducted using CPU by dividing the training data into chunks of 1000 sentences each. This chunk-based training allows the large dataset to be processed in incremented versions by preserving the optimizer and scheduler states through different iterations.

The proposed model gained an accuracy of 97.09%, precision of 97.02%, recall of 97.09%, and F1-score of 97.01%. These results outperform our baseline ML based Random Forest Model which was trained on hand crafted features and achieved accuracy of 91.2%. This proves the effectiveness of transformer-based fine-tuning for language identification in code-mixed contexts.

Index Terms **- DistilBERT, NLP, LID**

I. INTRODUCTION

Code-mixing means switching between two or more languages in same sentence or conversation. The use of code mixing is increasing as user created content is increasing day by day with availability of local language keyboards and AI assistants. This happens many often in casual online chats, comments, tweets and posts. Here, the language used by users is informal and there are no clear grammar rules defined. The users inserts the word or phrase to give more importance to some words or when they don't find an appropriate word to use in a language. This phenomenon brings significant challenges in NLP tasks.

The traditional approaches of language identification consisting dictionary lookups, rule based approaches and machine learning models often fail with lexical, phonetic and syntactic variations found in code mixed data which has no clearly defined rules. The newer transformer based models like BERT and its variants have shown major progress in the ability to capture and understand contextual dependencies. This make them ideal for complexities involved in code-mixed language.

This transformer based architectures requires computationally advanced resources like GPU to train. This puts a limit on its application in low resource set up. To mitigate this gap, we present an approach that fine tunes version of BERT called DistilBERT entirely on CPU. We divide the training data into manageable chunks of 1000 sentences and training the model incrementally. This division of data in to chunks allows a large sized dataset to be fine-tuned in a resource constrained CPU based environment.

For fine tuning dataset of 77,761 sentences is used, which consists of 7, 32, 197 words labeled as one of the three categories: Gujarati (GJ), English (EN), and Other (OT). The OT class was given to non-linguistic labels like named entities, numerals and special characters. Various YouTube channels having Gujarati background across diverse areas such as News, Movies, and Education, Cooking and social media influencers-motivational speakers were searched. We used Facepager application for scrapping these selected channels. The scrapped comments were manually reviewed for Gujarati-English code mixed sentences either in Gujarati script or Romanized script. We were left with 77,761 sentences which were cleaned and given for annotation. The annotation is performed by two annotators who were given Python-Telegram bot for the same. The sentences of the dataset are rich in their linguistic diversity and represents the real world code mixed pattern found in various online platforms.

Related Work

Gujarati is a low-resource language, existing techniques for language identification works well on monolingual or well-separated multilingual texts but fails in cod-mixed content. These techniques are based on statistical models, character-level n-grams, or dictionary-based matching with machine learning algorithms for classification.

Different researchers have applied rule-based and supervised learning techniques for word-level tagging. The algorithms like decision trees, conditional random fields (CRFs), and support vector machines (SVMs) with statistical and neural methods have been majorly used.

The deep learning models like recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and convolutional neural networks (CNNs) have led to noticeable improvement in language identification task. These deep learning models require huge amount of data and high-end computing resources, so their use in low resource environments in still a challenge.

The most recent transformer based architectures with BERT and its variants captures context more effectively and they outperform earlier models in several NLP tasks. For language identification and code mixed language modeling BERT variants like mBERT, XLM-R and DistilBERT have been widely used. Their fine tuning, requires use of GPU and large RAM for the state-of-the-art performance.

DistilBERT-a distilled version of BERT- has given balanced performance and efficiency particularly in resource-constrained environments. There have been very less exploration of using CPU for code mixed data in such models also. Our work fills this gap by exploring how chunk wise training on CPU can efficiently fine-tune DistilBERT for word-level language identification in Gujarati-English code-mixed text.

[1] Have conducted a survey on foundational for understanding computational code-mixing. They explored challenges, tasks, and techniques related to processing code-mixed language, including POS tagging, sentiment analysis, and language identification. They provided taxonomy and datasets used in earlier research and highlighted the need for tailored models.

[2] Have worked on bridging linguistic theory and computational modeling, introducing grammatical constraints to guide code-switching. Their study operationalizes theoretical constructs into features for machine learning models and shows how syntactic understanding improves code-mixing analysis.

[3] Have discussed how BERT revolutionized NLP by introducing deep bidirectional context modeling using transformers. It's pre-trained on large corpora and fine-tuned for downstream tasks. They noticed that BERT provides contextual embeddings that significantly improve tasks like NER, classification, and language

identification.

[4] Have discussed DistilBERT in their paper. It is a lighter version of BERT that maintains 97% of its performance while being 60% faster and smaller. It's useful for deployment in low-resource or compute-constrained environments and allows practical application of transformer models in real-time settings.

[5] Have evaluated the extent to which mBERT captures multilingual representations and cross-lingual transfer. They found that mBERT performs reasonably well even without explicit supervision for many languages.

[6] Have worked on improving automatic speech recognition (ASR) for Gujarati-English code-switched speech. They introduced methods to condition transformer layers on language identification at both word and character levels, enhancing the model's ability to distinguish between languages in spoken input.

[7] Have proposed a novel approach to spoken language identification (LID) by applying spectral augmentation techniques. Their method improves the robustness of LID systems in noisy environments, which is particularly beneficial for code-switched speech involving languages like Gujarati and English.

[8] Have introduced a Hierarchical Transformer (HIT) model, which is designed to capture the semantics of code-mixed languages. Evaluated across multiple Indian languages, including Gujarati, their model demonstrated superior performance in understanding the syntactic and semantic nuances of code-mixed text.

[9] Have assessed the code-mixing capabilities of multilingual large language models (LLMs) like GPT-3.5-turbo and GPT-4. They introduced rule-based prompting techniques to generate code-mixed sentences and evaluate the models' translation abilities across language pairs, including English-Gujarati.

[10] Have conducted a systematic review that analyzes various techniques for language identification in code-mixed text. It highlights challenges such as ambiguity, lexical borrowing, and intra-word code-mixing, providing a comprehensive overview of existing datasets and proposing a conceptual framework for future research.

[11] Have presented a linguistic resource for English-Gujarati code-mixed data, detailing their approach to language identification and normalization. Their work addresses the scarcity of resources for this language pair and contributes to the development of tools for processing code-mixed text.

[12] Have done a survey which provides an extensive overview of computational approaches to code-switched speech and language processing. They discussed challenges, existing tools, and future directions, offering valuable insights for researchers working on code-mixed language identification.

[13] Have conducted a study which explores machine learning techniques to predict code-switching points in bilingual text. Having understanding of where code-switching occurs can enhance language identification systems by providing contextual cues for language boundaries.

[14] Have addressed the challenges of language identification in code-mixed social media text. They presents a dataset and baseline systems for word-level language identification, highlighting the complexities introduced by informal and non-standard language use.

[15] Have explored inducing character-level noise during fine-tuning to enhance zero-shot cross-lingual transfer to dialects and closely related languages. The authors fine-tune BERT on sentence-level classification tasks and evaluate its performance on unseen dialects and languages, demonstrating that character-level noise can significantly aid cross-lingual transfer under certain conditions.

[16] Have investigated using integer arithmetic for both forward and backward propagation during the fine-tuning of BERT. The authors demonstrate that fine-tuning with 16-bit integer BERT matches the performance of 16-bit and 32-bit floating-point baselines, offering a more memory-efficient approach to fine-tuning.

[17] Have proposed the Freeze and Reconfigure (FAR) method, a memory-efficient training regime for BERT-like models that reduces memory usage during fine-tuning by avoiding unnecessary parameter updates. FAR reduces fine-tuning time and memory operations, making it suitable for resource-constrained devices.

[18] Have employed a transfer learning approach by applying a BERT model for word-level language identification in code-mixed Assamese-English text. The study finds that BERT outperforms traditional methods, achieving an accuracy of 94% in language identification tasks.

[19] Have explored multilingual BERT for language identification tasks and is relevant for the word-level language identification task you're working on. It shows how transformer-based models can be fine-tuned for multilingual tasks, including language identification.

[20] Have discussed how deep neural networks can be applied to word-level language identification tasks. It includes architectures similar to the ones you're interested in but focuses on traditional RNNs and CNNs. However, the principles can be adapted to transformers such as DistilBERT.

[21] Have explored how zero-shot learning can be applied to text classification tasks using pre-trained transformer models. While this is more about classification than word-level identification, it provides insight into how pre-trained models like BERT and DistilBERT can be leveraged without fine-tuning on labeled data.

[22] Have provided a broad overview of various transformer-based models and their applications across multiple NLP tasks, including language identification. It can help provide background on transformer models' impact and performance in language processing.

[23] Have shown techniques for fine-tuning BERT on Named Entity Recognition (NER) tasks with multilingual datasets, which may be relevant if your word-level language identification task involves multilingual data. It also provides practical approaches for fine-tuning transformer models.

Methodology

The main goal of this work was to develop an efficient and contemporary solution for word level language identification. For word-level language identification in Gujarati-English code-mixed text using a transformer-based architecture, but were faced with constraints like only CPU availability and limited RAM. Our methodology is based on four main components: selecting appropriate transformer model, dataset preprocessing and language mapping, chunk wise fine tuning and performance evaluation.

1.1 Model Selection

We were having limited availability of resources, so we searched a transformer model which is good at performance and also having less requirements. Our search resulted in adopting of DistilBERT, a distilled version of the BERT-base model, it retains 97% of BERT's language understanding capabilities while being 40% smaller and significantly faster. It is the candidate variant of BERT when we want to fine tune with CPU based environment.

DistilBERT is based on student-teacher framework which retains every second layer of BERT-base, resulting in only 6 transformer layers. Each layer contains multi-head self-attention and a feed-forward sub-layer with residual connections and layer normalization.

Why DistilBERT Architecture?

DistilBERT is a streamlined and optimized variant of BERT, created through a process known as knowledge distillation. Despite being more compact—40% smaller and significantly faster in inference (around 60%)—it retains approximately 97% of BERT's original performance in natural language understanding. Several core architectural modifications distinguish DistilBERT from the base BERT model:

- **Layer Reduction**: While BERT-base consists of 12 transformer blocks, DistilBERT compresses this to 6. These are initialized by selecting alternate layers from the original BERT checkpoint.
- **Hidden Dimensions**: The dimensionality of token embeddings remains unchanged at 768, preserving representational capacity.
- **Multi-Head Attention**: Each of the six layers retains 12 attention heads, just as in BERT-base.
- **Simplified Embedding Scheme**: DistilBERT omits token-type embeddings, which BERT employs for distinguishing between sentence pairs—this change simplifies the model, especially for single-sequence tasks.
- **Distillation-Based Training**: During its pre-training phase, DistilBERT is optimized using a combination of traditional masked language modeling loss and a distillation loss that encourages it to match the outputs of the larger teacher model (BERT).
- **Positional Representation**: Like BERT, DistilBERT uses learned positional embeddings to maintain word order and syntactic information.

Trained on the same datasets as BERT—the English Wikipedia and the Toronto Book Corpus—DistilBERT delivers near-parity in performance while offering better efficiency, making it ideal for scenarios with limited computational resources.

When using the Hugging Face Transformers library, the DistilBERT model can be easily loaded along with its tokenizer. This tokenizer leverages the WordPiece subword algorithm, which can cause discrepancies between original tokens and subword tokens—a mismatch that is carefully handled during model training and label alignment.

1.2 Dataset

The sentences curated by scrapping YouTube channels mentioned above resulted in 1,28,240 statements. The scrapped data consisted of not useful statements like statements other than Gujarati-English code mixed, only symbols and emojis etc. The statements written in Gujarati script if found useful were considered and transliterated to their roman equivalent. After careful screening we had collection of comprises 77,761 code-mixed Gujarati-English sentences, totaling 732,917 words.

Each word is identified as either Gujarati (GJ), English (EN), or Other (OT) — the latter representing named entities, special tokens, and numerals. The language distribution is as follows:

- Gujarati (GJ): 410,891 words
- English (EN): 269,600 words
- Other (OT): 52,035 words

Each sentence was preprocessed to extract individual words and their corresponding language tags. These words were then tokenized using DistilBERT tokenizer, with a maximum token length of 10 to account for short word lengths.

1.2.1 Text Preprocessing & Tokenization

We used Telegram-Python based boat to annotate dataset which was annotated at word level for each sentence by two annotators. For each word of a sentence the result of annotation was stored with word_lang format like hello_EN kem_GJ cho_GJ. Each sentence was processed to fetch all words and their corresponding tag. These words were then tokenized using DistilBERT tokenizer, we kept a maximum word length limit to 10 characters.

We used DistilBertTokenizerFast from Hugging Face Transformers library for tokenization of each word. To make the input suitable for model input, the tokenizer was configured with truncation and padding. The tokenization process ensures that every word is converted into a sequence of integers matching to the vocabulary of pre trained DistilBERT model.

1.2.2 Language Label Mapping

The language labels were mapped to integer values using a predefined mapping:

- GJ (Gujarati) was mapped to 1,
- EN (English) was mapped to 0,
- OT (Other) was mapped to 2.

This mapping converted the categorical language labels into numerical format, making them suitable for training in a classification setting.

1.3 Chunk wise fine-tuning

Having faced memory and computational resources constraints, we employed a chunk wise fine tuning strategy to fine-tune DistilBERT on CPU. We set chunk size to 1000 sentence per chunk, which allowed us incremental model training without need of sophisticated resources. We divided the original dataset into subset of 1000 sentences, trained the chunk, updated the model parameters and trained the next chunk. This allowed model to be fine-tuned on different portions of data and also through many sessions of training spanned through different days. The following flowchart shows the chunk wise fine tuning process.

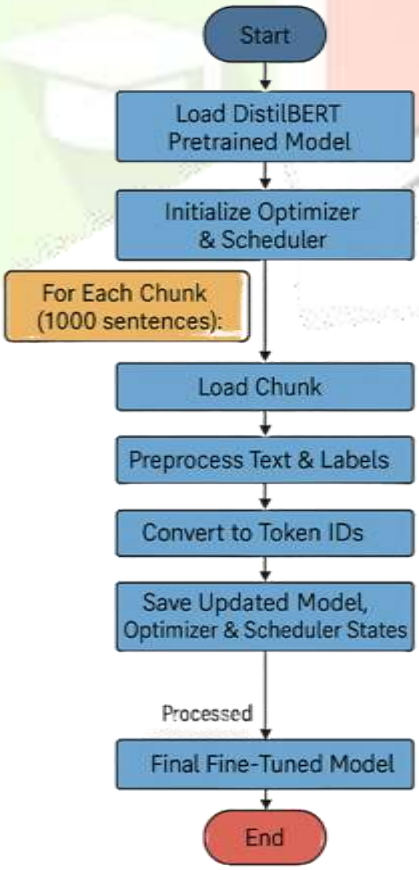

Figure 1

1.3.1 Model Initialization

We initialized the model with three distinct language labels of dataset EN, GJ, and OT. If we found a previously trained model checkpoint, then model was loaded from the directory checkpoint to resume training process. If the model checkpoint is not available, then a new model was initialized from the pre-trained DistilBERT uncased model.

1.3.2 Chunk Wise Training

To streamline the process if model training, evaluation and saving checkpoint, we used Hugging Face Trainer API. To train each chunk using iterative procedure, following was used.

A) We created a training dataset and validation dataset by tokenizing process of words and associating the correct language labels.

B) The Trainer API was used to initiate the training process. It was ensured that each chunk's training and validation datasets were properly loaded and passed to the trainer.

C) After training on each chunk, the model, tokenizer, and trainer state (optimizer and scheduler) were saved to the checkpoint directory for future use.

1.3.3 Model Saving

Once each chunk was trained, the results were saved which includes model weights, tokenizer configurations, and training states. This ensures that training could be resumed without data loss. This incremental approach on small datasets allows the model to learn from smaller portions of data, while benefitting from fine-tuning on domain specific language data.

1.3.4 Training Resumption

We kept a chunk index that allowed resuming of training process from previously saved checkpoint. This chunk index could be manually set thus making our process flexible and adaptable. This chunk-wise fine-tuning strategy efficiently uses the DistilBERT model's capabilities while managing computational resources effectively. The methodology ensures that the model is incrementally updated and that the entire dataset contributes to the final language identification model.

1.4 Performance evaluation

To assess the effectiveness of the chunk-wise CPU fine-tuning of DistilBERT for word-level language identification, we compared its performance with a RF (Random Forest) based model which was used as a baseline model. Our goal was to measure the model's ability to accurately identify the language tag of each word in real-world, informal digital communication.

1.4.1 Evaluation matrices

We adopted standard multi-class classification metrics to evaluate performance across the three language tags: gj (Gujarati), en (English), and ot (Other). The following metrics were computed:

- Accuracy: It predicts overall percentage of correctly predicted tags.
- Precision, Recall, F1-Score: It is calculated for each class and are macro-averaged to evaluate balanced performance across classes.
- Confusion Matrix: It is mainly applied analyze common misclassification patterns among the three labels.

1.4.2 Baseline model

The baseline model with RF algorithm was trained on 18 handcrafted linguistic and statistical features, including word length, script type, transliteration patterns, dictionary match, word frequency, sentiment indicators, and morphological properties. To ensure fair comparison with our fine-tuned model, this RF based model was trained on same dataset. The model achieved overall accuracy of 91.2%.

1.4.3 Fine-tuning results

The fine-tuned DistilBERT model achieved the following performance metrics on the test set:

- Accuracy: 97.09%
- Precision: 97.02%
- Recall: 97.09%
- F1-score: 97.01%

This is a significant improvement over the baseline RF+Handcrafted model, which achieved an F1-score of 91.2%. The result clearly shows improvement and proves that usage of contextualized embeddings and deep transformer architectures in dealing with code-mixed data is beneficial. The following diagram represents the confusion matrix for the same.

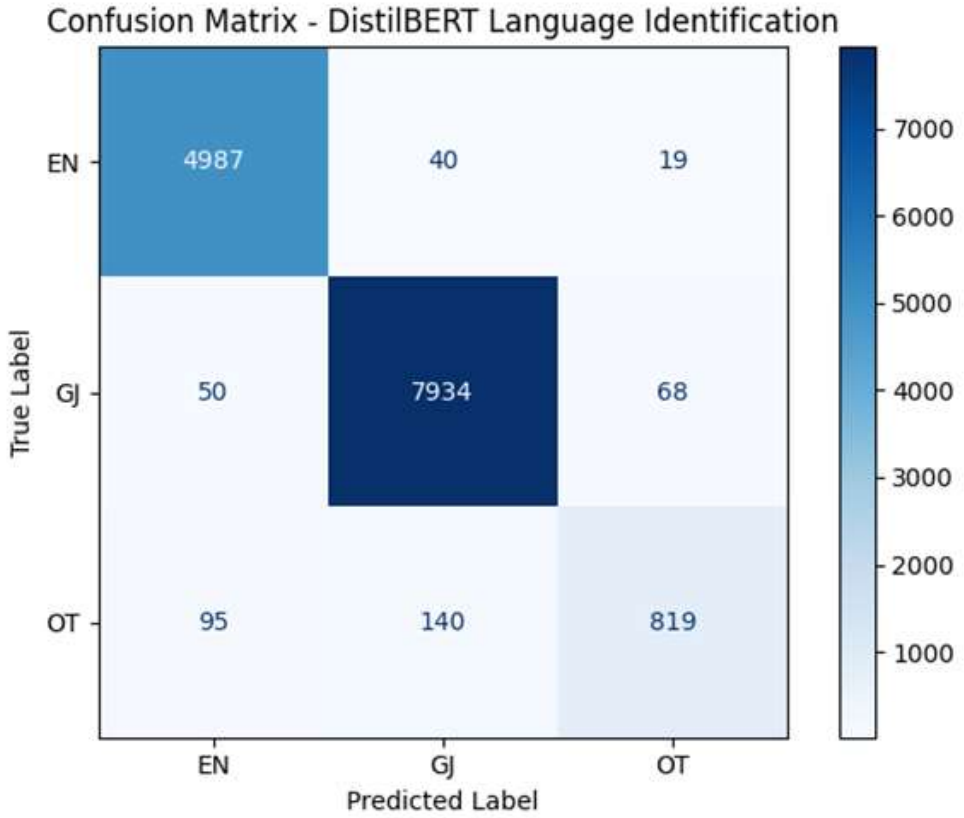

Figure 2

1.4.4 Comparison of results

The following table outlines the comparison of fine-tuned DistilBERT with RF+Handcrafted model on our evaluation matrices.

Table 1: Performance comparison with baseline model

Model	Precision	Recall	F1-score	Accuracy
RF With Handcrafted Features (Baseline)	91.27	91.27	91.27	91.27
DistilBERT Fine-tuning	97.02	97.09	97.01	97.09

Future Work

We found the following directions of to explore in our future work.

A) Handling imbalanced classes: We noticed that "OT" class labels were less as compared to the other two labels. To deal with this skewed class distribution, we would consider implementing over sampling techniques like SMOTE.

B) Applying multilingual transformers: We would use multilingual variants of transformer architectures like mBERT which can allow the model to take benefit from pre-existing knowledge of Indo-Aryan languages for better generalization.

C) Usage in downstream NLP applications: Combining the precise language identification information at word level into downstream NLP applications like sentiment analysis, named entity recognition, or machine translation for code-mixed texts can lead to better results and performance of this models.

References

1. Joshi, A., Bhattacharyya, P., & Carman, M. J. (2016). Investigations in computational code-mixing: A survey.

2. Bhat, R. A., et al. (2017). Grammatical constraints on intra-sentential code-switching: From theories to working models.

3. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding.

4. Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.

5. Pires, T., Schlinger, E., & Garrette, D. (2019). How multilingual is multilingual BERT?

6. Sharma, Y., Abraham, B., & Jyothi, P. (2024). Gujarati-English Code-Switching Speech Recognition using Ensemble Prediction of Spoken Language.

7. Rangan, P., Teki, S., & Misra, H. (2020). Exploiting Spectral Augmentation for Code-Switched Spoken Language Identification.

8. Sengupta, A., Suresh, T., Akhtar, M. S., & Chakraborty, T. (2022). A Comprehensive Understanding of Code-Mixed Language Semantics using Hierarchical Transformer.

9. Gupta, A., Bhogal, A., & Ghosh, K. (2024). Code-Mixer Ya Nahi: Novel Approaches to Measuring Multilingual LLMs' Code-Mixing Capabilities.

10. Rani, S., & Singh, M. (2023). A Systematic Review on Language Identification of Code-Mixed Text: Techniques, Data Availability, Challenges, and Framework Development.

11. Patel, R., & Shah, P. (2020). Language Identification and Translation of English and Gujarati Code-Mixed Data.

12. Sitaram, S., Chandu, K. R., Rallabandi, S. K., & Black, A. W. (2019). A Survey of Code-Switched Speech and Language Processing.

13. Solorio, T., & Liu, Y. (2008). Learning to Predict Code-Switching Points.

14. Barman, U., Das, A., Wagner, J., & Foster, J. (2014). Code Mixing: A Challenge for Language Identification in the Language of Social Media.

15. Aarohi Srivastava, David Chiang (2023). Fine-Tuning BERT with Character-Level Noise for Zero-Shot Transfer to Dialects and Closely-Related Languages

16. Mohammadreza Tayaranian Hosseini, Alireza Ghaffari, Marzieh S. Tahaei, et al. (2023) Towards Fine-tuning Pre-trained Language Models with Integer Forward and Backward Propagation

17. Danilo Vucetic, Mohammadreza Tayaranian, Maryam Ziaeefard, et al. (2022) Efficient Fine-Tuning of BERT Models on the Edge

18. Nayan Jyoti Kalita, Pritam Deka, Vijay Chennareddy, and Shikhar Kumar Sarma (2023). BERT Based Language Identification in Code-Mixed English-Assamese Social Media Text

19. A. Malik, A. Shrestha, C. W. T. Chang, P. S. Yan, A. T. Chan (2019). Language Identification in the Wild: A Case Study on the 13th International Workshop on Semantic Evaluation

20. A. Ghosal, S. Chakrabarti, S. Ghosal (2021). Word-Level Language Identification using Deep Neural Networks

21. K. Khandelwal, S. Tiwari. Zero-shot Text Classification with HuggingFace Transformers

22. C. Wolf, L. Debut, V. Sanh, T. Chaumond, J. Thomas, P. S. L. Martin. Transformers in Natural Language Processing: Applications and Challenges

23. D. Nguyen, P. Nguyen, H. Pham. Fine-Tuning BERT for Named Entity Recognition with Multilingual Datasets